SANDIA REPORT

SAND2018-3233
Unlimited Release
Printed March 28, 2018

Sierra/SolidMechanics 4.48 Goodyear Specific

SIERRA Solid Mechanics Team
Computational Solid Mechanics and Structural Dynamics Department
Engineering Sciences Center

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2018-3233
Unlimited Release
Printed: March 28, 2018

Sierra/SolidMechanics 4.48 Goodyear Specific

SIERRA Solid Mechanics Team
Computational Solid Mechanics and Structural Dynamics Department
Engineering Sciences Center
Sandia National Laboratories
Box 5800
Albuquerque, NM 87185-0380

Acknowledgments

This document is the result of the collective effort of many individuals. The current core development team responsible for the Sierra/SM codes includes Nathan K. Crane, Gabriel J. de Frias, San Le, David J. Littlewood, Mark T. Merewether, Matthew D. Mosby, Kendall H. Pierson, Julia A. Plews, Vicki L. Porter, Timothy R. Shelton, Jesse D. Thomas, Michael R. Tupek, Michael G. Veilleux, and Patrick G. Xavier. This document is written and maintained by this team.

Outside the core development team, there are many other individuals who have contributed to this manual. Nicole L. Breivik, J. Franklin Dempsey, Jeffery D. Gruda, Kevin N. Long, Kyran D. (Kim) Mish, and Chi S. (David) Lo have provided valuable input from the user community as Product Managers.

Many others have contributed to this document, either directly or indirectly. These include, but are not limited to Manoj K. Bhardwaj, Arthur Brown, James V. Cox, David M. Day, James W. Foulk III, Jason D. Hales, Brian D. Giffin, Steven P. Gomez, Arne S. Gullerud, Daniel C. Hammerand, Martin W. Heinstein, David M. Hensinger, Michael C. Hillman, Jason T. Ivey, Timothy D. Kostka, J. Richard Koteras, Alex J. Lindblad, Edward Love, Jakob T. Ostien, Rhonda K. Reinert, Nathaniel S. Roehrig, William M. Scherzinger, Gregory D. Sjaardema, Benjamin W. Spencer, Brian L. Stevens, Julia R. Walker, Gerald W. Wellman, and Edouard Yreux.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document covers Sierra/SolidMechanics capabilities specific to Goodyear use cases. Some information may be duplicated directly from the Sierra/SolidMechanics User's Guide [1] but is reproduced here to provide context for Goodyear-specific options.

References

[1] Sierra/SolidMechanics Team. Sierra/SolidMechanics 4.46 User's Guide. Technical Report SAND2017-9759, Sandia National Laboratories, Albuquerque, NM, 2017.

Chapter 2

Refer to User's Manual

Please refer to the Sierra/SolidMechanics User's Guide.

Implicit Only

Chapter 3

Implicit Solver Options

3.1 Control SST Root Finding

```
  BEGIN CONTROL SST ROOT FINDING
    TARGET RESIDUAL = <real>target
      [DURING <string list>period_names]
    TARGET RELATIVE RESIDUAL = <real>target_rel
      [DURING <string list>period_names]
    ACCEPTABLE RESIDUAL = <real>accept
      [DURING <string list>period_names]
    ACCEPTABLE RELATIVE RESIDUAL = <real>accept_rel
      [DURING <string list>period_names]
    LEVEL = <integer>stiffness_level
    MINIMUM ITERATIONS  = <integer>min_iter(0)
      [DURING <string list>period_names]
    MAXIMUM ITERATIONS  = <integer>max_iter
      [DURING <string list>period_names]
    REFERENCE = EXTERNAL|INTERNAL|BELYTSCHKO|RESIDUAL|ENERGY(EXTERNAL)
      [DURING <string list>period_names]
    RESIDUAL NORM TYPE = All|Scale_RB_Rotations|Translation(ALL)
    RESIDUAL ROUNDOFF TOLERANCE = <real>round_tool(1.0e-15)
      [DURING <string list>period_names]
  END
```

Stead state transport (SST) is a way to represent rotating bodies without explicit mesh rotation. SST involves material flowing through the mesh and generally should only be applied to rotational symmetric bodies. The `CONTROL SST ROOT FINDING` command block defines the solver options used to process `STEAD STATE TRANSPORT` boundary conditions.

The tolerance definition commands for the SST root finding block function the same as the equivalent commands in the `CG` solver block.

Note, the SST control should generally be applied by itself at the highest solver level. Thus if control contact or control stiffness are being used these controls should operate at level one and control SST root finding at level 2. As with other multilevel controls the higher controls should have significantly large convergence residuals than lower controls.

3.2 Control Stiffness

Control stiffness is a multi-level augmented Lagrange iterative solution strategy for solving mechanics models that have widely varying stiffness in various modes of material response. For instance, nearly incompressible materials have a bulk/volumetric response that is much stiffer than the corresponding shear/deviatoric response. A similar case arises for materials that are orthotropic or anisotropic in nature and exhibit widely varying stiffness in the principal material directions.

Using control stiffness allows part of the constitutive response of a material to be softened or stiffened to give a tangent response that results in model problems that can be solved more easily by the core solver. This is accomplished by scaling a chosen stress increment up or down in a given model problem and adding it to a saved reference stress to yield a scaled stress that is used for equilibrium evaluations. The reference stress accumulates the model problem stress increments so that the true material response for a time step is obtained after solving a sequence of model problems.

For the case of nearly incompressible materials, a sequence of model problems can be constructed in which the bulk behavior is softened (scaled down) and/or the shear behavior is stiffened (scaled up). For the case of anisotropic materials, the anisotropic part of the material response is scaled down to create a model problem where the material has a more nearly isotropic response. In addition, the isotropic part of the response of some of these orthotropic materials can have its bulk stress increments scaled down and/or its shear/deviatoric stress increments scaled up. Finally, if an isotropic material is much stiffer than its neighbors, all of that material's stress increments can be scaled down.

Scaling a stress increment up or down is akin to using scaled moduli in a model problem. This is only precisely true for the case in which the true material response is linear. For the case of nonlinear materials, the effective scaled moduli are based solely on the difference between the unscaled stresses and the reference stresses. As a result, these effective scaled moduli vary over the course of model problem and core solver iterations. Nevertheless, the control stiffness algorithms are completely general and do not rely on linearity in the true material response.

The degree to which components of a material's response are softened or stiffened to achieve overall minimum computational time is problem dependent and involves a trade-off between the difficulty of solving a model problem and the number of model problems that must be used to achieve the final solution.

For example, consider a nearly incompressible material, such as rubber, for which the model problem solution may be optimized by scaling the bulk and shear behaviors such that the effective tangent response has a bulk modulus that is equal to twice its shear modulus. Adjusting the ratio between the bulk and shear moduli in this manner may minimize the core solver iterations for a given model problem. However, because of the inherently large difference between these two moduli in this material, significant bulk and/or shear scaling would be required to achieve these optimal scaled moduli. Such severe scaling could result in a large number of model problems to achieve the true material response. Choosing less severe scalings (scalings closer to 1), on the other hand, could result in more core solver iterations in each model problem, but fewer model problems to arrive at the true material response.

Experience has shown that once nearly optimal scaling values have been determined for a class of problems, these can be applied successfully to families of problems with similar characteristics.

A partial explanation of scaling the material response will be given for a simple scalar relation between stress and strain. For example, the true pressure-volume relation for a nearly incompressible material is a scalar relation between pressure and volumetric strain. Appropriate generalizations of the softening and stiffening algorithms to tensor relations are easily achieved by applying the same algorithms for all of the stress quantities individually. Only the calculation of the appropriate error measures is modified to deal with the tensorial nature of the stress-strain response that is being scaled.

There are two strain and three stress quantities of interest in the control stiffness algorithm. The stress quantities are the unscaled stress calculated by the unmodified material model, the scaled stress used for equilibrium evaluation, and a reference stress used to build up stresses to achieve convergence such that the scaled model problem stress is nearly equal to the unscaled stress computed from the material model. The strain quantities of interest are the true strain computed from the kinematics and the strain necessary to achieve the scaled stress using the unmodified material model.

In the equations that follow, the subscripts refer to the particular model problem being solved. Let us begin with the true material response in model problem I given by

$$\sigma_I = f(e_I), \tag{3.1}$$

where σ_I is the true stress corresponding to the kinematic strain e_I determined from the nodal displacements, and $f(\cdot)$ is the function representing the constitutive response. Although the true material response has been written for the hyperelastic case employing total stress and strain quantities, the control stiffness algorithms that are being presented can be equally applied to hypoelastic models where the stress rate is written in terms of the strain rate. The important point here is that the unscaled constitutive equation is used to calculate the unscaled/true stress response. It is not necessary for that relation to be linear or for a particular formulation to be used in terms of total strains or incremental strain rates. However, it should be noted that control stiffness is set up to work only with certain material models.

The scaled stress response to use for equilibrium evaluations in a model problem is determined as follows:

$$s_I = r_I + \lambda \left(\sigma_I - r_I\right), \tag{3.2}$$

where s_I is the scaled stress, r_I is the reference stress, and λ is the scaling factor. For softening behavior, λ is less than 1, while for stiffening behavior, it is greater than 1. As noted previously, this scaled stress is what the core solver uses for equilibrium evaluation in a given model problem. The core solver typically requires many iterations to find the scaled stress that results in equilibrium. The softening and stiffening control stiffness algorithms differ only in terms of what is used for the reference stress. The reference stress in model problem I is given by the following:

$$r_I = \begin{cases} s_{I-1} & \lambda < 1 \quad \text{(softening)} \\ \sigma_{I-1} & \lambda > 1 \quad \text{(stiffening)} \end{cases} \tag{3.3}$$

Figures 3.1 and 3.2 provide a graphical presentation of softening the material response, and Figures 3.3 and 3.4 offer a similar presentation of stiffening the material response.

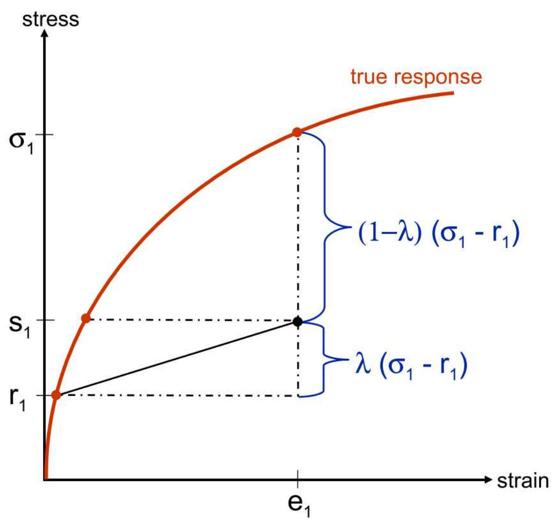

Figure 3.1: Control stiffness softening behavior in the first model problem of a time step.

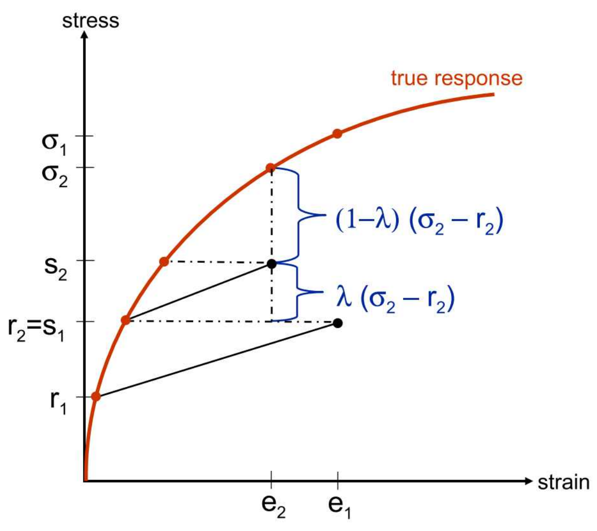

Figure 3.2: Control stiffness softening behavior in the second model problem of a time step.

When equilibrium is achieved in a model problem, it is necessary to do two things. First, an error measure related to the difference between the unscaled and scaled stress must be determined. Second, the reference stress used in the scaled stress calculations must be updated. The reference stresses are updated according to Equation (3.3). That is, for the case of softening the material response, the reference stress is updated for the next model problem to be the scaled stress that achieved equilibrium in the current model problem. On the other hand, for the case of stiffening the material response, the reference stress is updated for the next model problem to be the true stress at the end of the current model problem.

The control stiffness updating process is converged when the scaled and unscaled stresses differ by

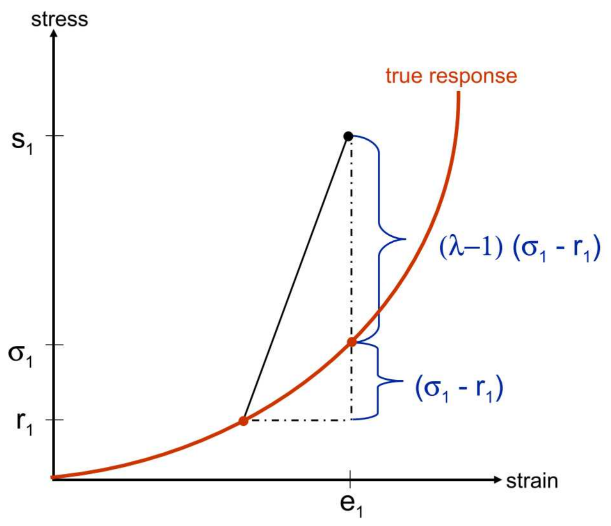

Figure 3.3: Control stiffness stiffening behavior in the first model problem of a time step.

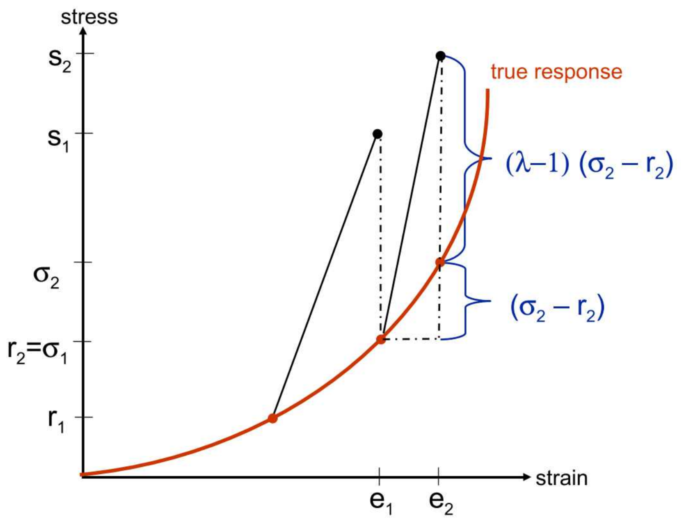

Figure 3.4: Control stiffness stiffening behavior in the second model problem of a time step.

an acceptably small amount. Several different error measures can be used to determine the degree to which the scaled and unscaled stresses differ. These can be categorized into two types: those that consider stress errors, and those that consider strain errors.

The stress convergence measures consider consider the difference between the scaled stress and the unscaled stress. The first of these stress error measures involves taking the L^2 norm of the stress difference:

$$error_I = \left\| s_I - \sigma_I \right\|_2 . \tag{3.4}$$

A second stress error measure involves normalizing that quantity:

$$error_I = \frac{\left\| s_I - \sigma_I \right\|_2}{\left\| s_I \right\|_2}. \tag{3.5}$$

The strain error convergence measure is based on a strain quantity E_I that is computed as the difference between the scaled stress and the unscaled stress and dividing that value by a modulus:

$$E_I = \frac{\left| s_I - \sigma_I \right|}{M}, \tag{3.6}$$

where M is an appropriate material modulus that is used to represent the unscaled constitutive response. For the case where the true material response is linear and the scaled stress is computed by stiffening the material response, E_I as defined by Equation (3.6) corresponds exactly to the difference between the kinematic strain e_I and the strain ϵ_I needed in the unscaled constitutive equation to give the scaled stress:

$$\epsilon_I = f^{-1}(s_I) = s_I/M, \tag{3.7}$$

where M would be the real material modulus. Nevertheless, Equation (3.6) is not dependent upon linear behavior, and it can be used in both softening and stiffening types of control stiffness scaling. The strain error quantity used to check convergence is computed by taking a global maximum (an L^∞ norm) as follows:

$$error_I = \left\| \frac{E_I}{E_{ref}} \right\|_\infty, \tag{3.8}$$

where E_{ref} is a reference strain specified in the input parameters for the material whose response is being scaled.

In summary, the user may specify error tolerances to determine convergence of the sequence of model problems solved in the control stiffness algorithm by using one of the error measures defined by Equations (3.4), (3.5), or (3.8). If the user so chooses, error tolerances corresponding to both Equations (3.4) and (3.5) can be specified, and convergence is considered whenever either tolerance is met. Because L^2 norms are computed by summing over all the elements in a mesh, users are cautioned that the tolerance used with Equation (3.4) to achieve a given level of control stiffness convergence is mesh dependent. That is, as the number of elements greatly increases, the L^2 norm of Equation (3.4) naturally increases. In most cases, the relative strain error given in Equation (3.8) is the preferred error measure. In addition to the error measures defined by Equations (3.4), (3.5), or (3.8), it is necessary to determine whether equilibrium is still achieved once the reference stresses have been updated. That is, equilibrium is re-evaluated with σ_{I+1} and s_{I+1} calculated without any additional changes to the fundamental nodal degrees of freedom such that

$$\sigma_{I+1} = \sigma_I \tag{3.9}$$

and

$$s_{I+1} = r_{I+1} + \lambda \left(\sigma_{I+1} - r_{I+1} \right). \tag{3.10}$$

If convergence is achieved both for the difference between the scaled stress and the unscaled stress (either as a direct measure or as a strain error) and for the updated equilibrium evaluation, the time step is considered complete. If control contact is also active, it is necessary that the appropriate contact convergence checks be satisfied as well.

Figure 3.5 shows an example of a linear material that has been softened through control stiffness. In this example, the time step is considered converged immediately after the second model problem update. The third model problem involved no iterations and is thus simply a re-evaluation of equilibrium and control stiffness convergence after the reference stress is updated following the second model problem.

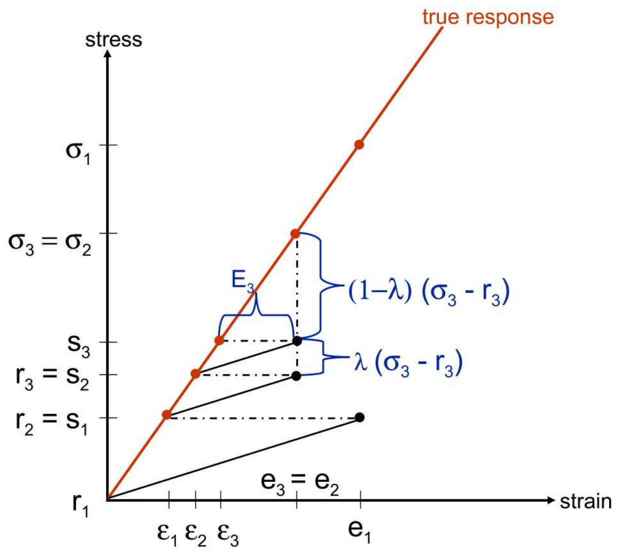

Figure 3.5: Control stiffness softening behavior convergence is achieved after solving two model problems.

The command block for control stiffness is as follows:

```
BEGIN CONTROL STIFFNESS [<string>stiffness_name]
  #
  # convergence commands
  TARGET <string>RESIDUAL|AXIAL FORCE INCREMENT|
    PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT
    = <real>target [DURING <string list>period_names]
  TARGET RELATIVE <string>RESIDUAL|AXIAL FORCE INCREMENT|
    PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT|
    STRAIN INCREMENT
    = <real>target_rel [DURING <string list>period_names]
  ACCEPTABLE <string>RESIDUAL|AXIAL FORCE INCREMENT|
    PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT
    = <real>accept [DURING <string list>period_names]
  ACCEPTABLE RELATIVE <string>RESIDUAL|AXIAL FORCE INCREMENT|
    PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT|
    STRAIN INCREMENT
    = <real>accept_rel [DURING <string list>period_names]
  REFERENCE = EXTERNAL|INTERNAL|BELYTSCHKO|RESIDUAL|
    ENERGY(EXTERNAL) [DURING <string list>period_names]
  MINIMUM ITERATIONS  = <integer>min_iter(0)
    [DURING <string list>period_names]
  MAXIMUM ITERATIONS  = <integer>max_iter
    [DURING <string list>period_names]
  #
  # level selection command
  LEVEL = <integer>stiffness_level
  #
  # diagnostic output commands, off by default.
  ITERATION PLOT = <integer>iter_plot
    [DURING <string list>period_names]
  ITERATION PLOT OUTPUT BLOCKS  = <string list>plot_blocks
END [CONTROL STIFFNESS <string>stiffness_name]
```

To enable control stiffness, a `CONTROL STIFFNESS` command block must exist in the `SOLVER` command block. The command lines within the `CONTROL STIFFNESS` command block are used to control convergence during the control stiffness updates, select the level for control stiffness within the multilevel solver, and output diagnostic information. These command lines are described in detail in Sections 3.2.1 through 3.2.3.

3.2.1 Convergence Commands

The command lines listed in this section are placed in the `CONTROL STIFFNESS` command block to control convergence criteria for stiffness control within the multilevel solver.

```
TARGET <string>RESIDUAL|AXIAL FORCE INCREMENT|
  PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT
  = <real>target [DURING <string list>period_names]
TARGET RELATIVE <string>RESIDUAL|AXIAL FORCE INCREMENT|
  PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT|
  STRAIN INCREMENT
  = <real>target_rel [DURING <string list>period_names]
ACCEPTABLE <string>RESIDUAL|AXIAL FORCE INCREMENT|
  PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT
  = <real>accept [DURING <string list>period_names]
ACCEPTABLE RELATIVE <string>RESIDUAL|AXIAL FORCE INCREMENT|
  PRESSURE INCREMENT|SDEV INCREMENT|STRESS INCREMENT|
  STRAIN INCREMENT
  = <real>accept_rel [DURING <string list>period_names]
REFERENCE = EXTERNAL|INTERNAL|BELYTSCHKO|RESIDUAL|ENERGY(EXTERNAL)
  [DURING <string list>period_names]
MINIMUM ITERATIONS = <integer>min_iter(0)
  [DURING <string list>period_names]
MAXIMUM ITERATIONS = <integer>max_iter
  [DURING <string list>period_names]
```

Convergence for problems using control stiffness requires that two criteria be met. The first criterion is that equilibrium must be achieved. The second criterion is that the scaled stress used for equilibrium must be close enough to the unscaled stress that is determined with the unmodified material model as calculated from either Equation (3.4), (3.5), or (3.8).

The command lines listed above for the equilibrium evaluation are similar to those used for the core CG solver, and have similar meaning. Here, however, the command lines are applied as part of determining convergence for the control stiffness series of model problems. Some of the command lines have multiple options; for instance, the command line beginning with `TARGET` has five options. This practice has been used for the command lines beginning with `TARGET`, `TARGET RELATIVE`. `ACCEPTABLE`, and `ACCEPTABLE RELATIVE`. Note also that all of these command lines can be appended with the `DURING` specification, as discussed in Section 2.

Convergence of equilibrium using scaled stresses is measured by computing the L^2 norm of the residual, and comparing that residual norm with target convergence criteria specified by the user. Convergence can be monitored either directly in terms of the residual norm, or in terms of a relative residual, which is the residual norm divided by a reference quantity that is indicative of the current loading conditions on a model. Either the residual, the relative residual, or both of these can be used for checking convergence at the same time. The `TARGET RESIDUAL` command line specifies the target convergence criterion in terms of the actual residual norm. The `TARGET RELATIVE RESIDUAL` command line specifies a convergence criterion in terms of the relative residual. If both absolute and relative criteria are specified for the residual calculation, the equilibrium is accepted if either criterion is met.

The multilevel solver also allows acceptable convergence criteria to be input for residual convergence. If the solution has not converged to the specified targets before the maximum number of iterations is reached, the residual is checked against the acceptable convergence criteria. These criteria are specified via the `ACCEPTABLE RESIDUAL` and `ACCEPTABLE RELATIVE RESIDUAL` command lines. The concepts of absolute and relative values are the same here as discussed for the target limits.

If relative residuals are given in the convergence criteria, the `REFERENCE` command line can be used to select the method for computing the reference load. This command line has several options: `EXTERNAL`, `INTERNAL`, `BELYTSCHKO`, `RESIDUAL` and `ENERGY`. When the `EXTERNAL` option, which is the default, is selected, the reference load is computed by taking the L^2 norm of the current external load. The `INTERNAL` option uses the norm of the internal forces as the reference load. The `BELYTSCHKO` option uses the maximum of the norm of the external load, the norm of the reactions, and the norm of the internal forces. The `RESIDUAL` option denotes that the residual from the initial residual should be used as the reference quantity. The `ENERGY` residual is the energy conjugate norm of the residual and reference vectors. These criteria are further explained in Section 2.

For control stiffness, unlike other controls, the convergence of the series of model problems to obtain the final solution for a time step is also based on other criteria that measure how far apart the scaled and unscaled stress responses are from each other. Either a direct error measure of the difference of the scaled and unscaled stress is used or a relative strain error computed using the difference between the scaled and unscaled stress and a representative modulus is employed.

There are a number of variants of the TARGET command that can be used for the direct stress differences. The TARGET AXIAL FORCE INCREMENT command is used to base convergence on the increment of axial force in fiber membrane elements in an update. The TARGET PRESSURE INCREMENT and TARGET SDEV INCREMENT commands are used to specify convergence for nearly incompressible materials based on the pressure and deviatoric stress increments, respectively. Also, TARGET STRESS INCREMENT is used to base convergence on the stress increment for materials which have their entire behavior softened as part of a control stiffness approach. If desired, the RELATIVE form of these convergence criteria can be used. If the TARGET RELATIVE STRAIN INCREMENT command is specified, the strain error measure between the scaled and unscaled stress will be computed and used.

Any combination of these criteria can be specified. If more than one is specified, all of the criteria must be satisfied. However, each material block contributes either to the direct stress error measures or to the relative strain error measure. If a material block specifies a positive non-zero REFERENCE STRAIN in its definition, it will contribute to the relative strain error measure. Otherwise, it will contribute to the direct stress error measures. Finally, acceptable convergence criteria can be input for converging the difference between the scaled and unscaled stress. If the solution has not met the target criteria for equilibrium and the difference between the scaled and unscaled stresses, but meets the acceptable criteria, it is allowed to proceed to the next time step.

The MAXIMUM ITERATIONS and MINIMUM ITERATIONS command lines specify the maximum and minimum number of control stiffness updates, respectively. The default minimum number of iterations, min_iter, is 0. If a number greater than 0 is specified for min_iter, the multilevel solver will update stiffness at least that many times, regardless of whether the convergence criteria have been met.

3.2.2 Level Selection Command

```
LEVEL = <integer>stiffness_level
```

The LEVEL command line in the CONTROL STIFFNESS command block is used to specify the level in the multilevel solver at which stiffness is controlled. The stiffness_level parameter can have a value of either 1 or 2, with 1 being the default.

This command is used when multiple controls (e.g., control contact and control stiffness) are active in the multilevel solver. It is permissible for multiple controls to exist at a given level. The default behavior is for all controls to be at level 1. To have a control at level 2, another control must be active at level 1 because a level in the multilevel solver cannot be skipped. The level of other controls is specified using the LEVEL command line in the command blocks associated with those controls.

3.2.3 Diagnostic Output Commands

```
ITERATION PLOT = <integer>iter_plot
  [DURING <string list>period_names]
ITERATION PLOT OUTPUT BLOCKS  = <string list>plot_blocks
```

The command lines listed above can be used to control the output of diagnostic information from the stiffness control within the multilevel solver. The `ITERATION PLOT` command line allows plots of the current state of the model to be written to the output database during the stiffness control updates. The value supplied in `iter_plot` specifies the frequency of such plots. The default behavior is not to produce such plots. The plots are generated with a fictitious time step. Every time such a plot is generated, a message is sent to the log file. This feature is useful for debugging, but it should be used with care. The `ITERATION PLOT` command line can be appended with the `DURING` specification, as discussed in Section 2.

The output produced by iteration plots is sent by default to every active results output file, as specified by a `RESULTS OUTPUT` command block. The `ITERATION PLOT OUTPUT BLOCKS` command line can optionally be used to supply a list of output block names to which iteration plots should be written. Each of the output block names listed in `plot_blocks`, must match the name of a `RESULTS OUTPUT` command block (see Section 2). The `ITERATION PLOT OUTPUT BLOCKS` command line can be useful for directing iteration plots to separate output files so that these plots are separated from the converged solution output. It can also be used to send output plots for different levels of the multilevel solver to different files.

3.3 Control Modes

The core conjugate gradient solution algorithm typically works very well on "blocky" problems if the nodal preconditioners are used. As the aspect ratios of the structures modeled increase, however, the performance of the CG solver begins to degrade. Because of their lack of coupling terms, the nodal preconditioners only allow for the effects of the residual to be propagated across a single element during an iteration. Consequently, these preconditioners are effective at minimizing high frequency error but may require many iterations to solve for the low frequency, structural bending modes in slender structures.

The full tangent preconditioners provided by Adagio greatly improve the effectiveness of the CG algorithm for solving models with slender members. However, the memory usage requirements and the computational effort needed to factorize matrices with the full tangent preconditioners can sometimes be significant.

In addition to its other solution strategies, Adagio provides a multigrid solution method within the CG algorithm. This method, known as control modes, greatly increases the effectiveness of the CG algorithm for solving slender, bending-dominated problems. In addition to the actual mesh of the model to be solved, called the "reference mesh," control modes uses a coarse mesh of the model, known as a "constraint mesh," to solve for the low-frequency response. To use control modes, the user must supply both the reference mesh and the constraint mesh. While it does require some additional effort to generate a suitable coarse mesh, control modes provides efficient solutions with only slightly higher memory usage than would be required by the standard CG solver with a nodal preconditioner.

Control modes functions somewhat like another level in the multilevel solver, although it does not appear as a control type in the `SOLVER` command block. The CG algorithm operates alternately on the residual on the coarse mesh and the fine mesh. The coarse residual is computed by performing a restriction operation from the fine mesh to the coarse mesh using the shape functions of the coarse elements. The CG solver is used to minimize the coarse residual until convergence is reached, at which point it switches to operate on the fine mesh to minimize that residual. The fine mesh iterations continue until either the fine mesh has converged or the fine residual is 50% of the coarse residual. The solver alternates between the fine and coarse meshes until convergence is obtained on both meshes.

To use control modes, the user should set up the mesh file and the input file as usual, except that the following additional items must be provided:

- A constraint mesh must be generated. The constraint mesh must be in a separate file from the reference mesh, which is the real model. The constraint mesh should be a coarse representation of the reference mesh. If there are node sets or side sets in the reference mesh that are used to prescribe kinematic boundary conditions, similar mesh entities should be provided in the coarse mesh to prescribe similar boundary conditions.
- A second `FINITE ELEMENT MODEL` command block must be provided in addition to the standard definition for the reference finite element model in the input file. This command block is set up exactly as it normally would be (see Section 2), except that the mesh file referenced is the constraint mesh instead of the reference mesh. Although the constraint mesh is used purely as a solution tool, and does not use any finite elements or material

models, each block in the constraint mesh must still be assigned a material model.

- A CONTROL MODES REGION command block must appear alongside the standard ADAGIO REGION command block within the ADAGIO PROCEDURE command block. The presence of the CONTROL MODES REGION command block instructs the CG solver to use the control modes logic. There are no commands within the CG solver for the region associated with the reference mesh related to control modes. The CONTROL MODES REGION command block is documented in Section 3.3.1. It contains the same commands used within the standard ADAGIO REGION command block, except that the commands in the CONTROL MODES REGION command block are used to control the control modes algorithm and the boundary conditions on the coarse mesh.

3.3.1 Control Modes Region

```
BEGIN CONTROL MODES REGION
  #
  # model setup
  USE FINITE ELEMENT MODEL <string>model_name
  CONTROL BLOCKS = <string list>control_blocks
  #
  # solver commands
  BEGIN SOLVER
    BEGIN LOADSTEP PREDICTOR
      #
      # Parameters for loadstep predictor
      #
    END [LOADSTEP PREDICTOR]
    BEGIN CG
      #
      # Parameters for CG
      #
    END [CG]
  END [SOLVER]
  JAS MODE [SOLVER|CONTACT|OUTPUT]
  #
  # kinematic boundary condition commands
  BEGIN FIXED DISPLACEMENT
    #
    # Parameters for fixed displacement
    #
  END [FIXED DISPLACEMENT]
  BEGIN PERIODIC
    #
    # Parameters for periodic
    #
  END [PERIODIC]
END [CONTROL MODES REGION]
```

The CONTROL MODES REGION command block controls the behavior of the control modes algorithm and is placed alongside a standard ADAGIO REGION command block within the ADAGIO PROCEDURE scope. With the exception of the CONTROL BLOCKS command line, all the commands that can be used in this block are standard commands that appear in the Adagio region. These commands have the same meaning in either context; they simply apply to the constraint mesh or to the reference mesh, depending on the region block in which they appear. Sections 3.3.1.1 through 3.3.1.3 describe the components of the CONTROL MODES REGION command block.

3.3.1.1 Model Setup Commands

```
USE FINITE ELEMENT MODEL <string>model_name
CONTROL BLOCKS = <string list>control_blocks
```

The command lines listed above must appear in the CONTROL MODES REGION command block if control modes is used. The USE FINITE ELEMENT MODEL command line should reference the finite element model for the constraint mesh. This command line is used in the same way that the command line is used for the reference mesh (see Section 2).

The CONTROL BLOCKS command line provides a list of blocks in the reference mesh controlled by the constraint mesh in the solver. The block names are listed using the standard method of referencing mesh entities (see Section 2). For example, the block with an ID of 1 would be listed as block_1 in this command. Multiple CONTROL BLOCKS command lines may appear to specify a long list of blocks over several lines.

3.3.1.2 Solver Commands

```
BEGIN SOLVER
  BEGIN LOADSTEP PREDICTOR
    #
    # Parameters for loadstep predictor
    #
  END [LOADSTEP PREDICTOR]
  BEGIN CG
    #
    # Parameters for CG
    #
  END [CG]
END SOLVER
JAS MODE [OUTPUT|CONTACT|SOLVER]
```

The constraint mesh must have solver parameters defined using the LOADSTEP PREDICTOR and CG command blocks, which must be nested within a SOLVER block in the CONTROL MODES REGION. The constraint mesh does not have material properties or contact, so the CG command block is used to define the solver on the constraint mesh. The multilevel solver is not used on the constraint mesh, even though the multilevel solver may be used on the reference mesh in an analysis that uses a constraint mesh. All the command lines that appear in these command blocks in a standard analysis can be used for the parameters of the solver on the constraint mesh. One additional solver detail is that "intermediate convergence" will be indicated with `:C` on the CG solver's final step. The solution will then be updated and the solver will continue with the next alternation of reference and constraint. Refer to Section 2 for a description of the command lines for the load step predictor and to Section 2 for a description of the CG solver commands.

The full set of preconditioners that are available on the reference mesh are also available for the constraint mesh. The full tangent preconditioner can be used on the constraint mesh, and provides a powerful solution strategy. Because the constraint mesh is typically small compared to the reference mesh, forming and factorizing matrices for the constraint mesh typically requires small computational resources.

The `JAS MODE` command line enables complete compatibility with the JAS3D legacy code. If this mode of operation is desired, the `JAS MODE` command line must be present in both the `CONTROL MODES REGION` command block and the `ADAGIO REGION` command block. See Section 3.4 for more information on this command line.

Warning: The `RESIDUAL NORM TYPE` command in the CG command block should not be used in a `Control Modes` regions.

3.3.1.3 Kinematic Boundary Condition Commands

```
BEGIN FIXED DISPLACEMENT
  #
  # Parameters for fixed displacement
  #
END [FIXED DISPLACEMENT]
BEGIN PERIODIC
  #
  # Parameters for periodic
  #
END [PERIODIC]
```

For the solver to converge well, it is often important to create a node set or a side set on the coarse mesh that coincides geometrically with a node set or a side set on the fine mesh to which a fixed boundary condition is applied. This should be done for fixed boundary conditions, but not for boundary conditions with prescribed nonzero displacements. Any of the kinematic boundary condition command blocks that can appear in a standard Adagio region can also appear in the `CONTROL MODES REGION` command block. However, it is recommended that only `FIXED DISPLACEMENT` and `PERIODIC` boundary conditions be used on the constraint mesh.

The boundary conditions are applied to the specified mesh entities on the constraint mesh. The constraint mesh and the reference mesh can have a node set or a side set with the same identifier. The determination of which entity should be affected by the boundary condition is based on the context of the boundary condition specified. See Section 2 for details on the `FIXED DISPLACEMENT` command block.

3.4 JAS3D Compatibility Mode

Adagio's multilevel solver and CG solver are based on the solver used in the legacy JAS3D code [1]. While the solvers in the two codes are similar, there are some minor implementation differences between the two codes.

```
    JAS MODE [SOLVER|CONTACT|OUTPUT]
```

Adagio provides an option, selectable by inserting the `JAS MODE` command line in the `ADAGIO REGION` command block, to make it use exactly the same algorithms as the JAS3D solver. If control modes is being used, the `JAS MODE` command line must appear both in the `ADAGIO REGION` and in the `CONTROL MODES REGION` command blocks (see Section 3.3.1.2). This option is primarily useful for migrating analyses previously done in JAS3D to Adagio.

The `JAS MODE` command used without any options enables all JAS3D compatibility features available. Optionally, this command can be followed by one of the three keywords `SOLVER`, `CONTACT`, or `OUTPUT` to enable a subset of these features. The command can be repeated on separate lines with different options to enable more than one subset of features.

The subsets of the features enabled by the `JAS MODE` command line are summarized below:

- `SOLVER`: Adagio's CG solver and multilevel solver use exactly the same algorithms. There are many algorithmic decisions in the solver that were made one way in JAS3D and another way in Adagio. While both methods are valid, the `JAS MODE` command line forces the code to always use the exact JAS3D algorithms.
- `CONTACT`: Adagio uses the Legacy Contact library instead of the internal Adagio and ACME code for contact search and enforcement. The Legacy Contact library is the same contact code that is used in JAS3D, so contact in Adagio models behaves the same as it would if it were run in JAS3D if this option is chosen.
- `OUTPUT`: The log file output produced by Adagio is formatted in the same manner as JAS3D formats log files, thus facilitating comparisons between the output from JAS3D and Adagio models.

The `JAS MODE` command line only ensures JAS3D compatibility for features that are available in JAS3D. Advanced features such as the full tangent preconditioner are not available in JAS3D. It should also be noted that for those interested in converting JAS3D input files to Adagio input files, there is a program available. For more information on this converter program, refer to the Adagio web page or contact an Adagio developer.

References

[1] M.L. Blanford, M.W. Heinstein, and S.W. Key. JAS3D – a multi-strategy iterative code for solid mechanics analysis users' instructions, release 2.0, September 2001.

Chapter 4

Materials

This chapter describes material models specific to Goodyear.

4.1 Fiber Membrane Model

```
BEGIN MATERIAL <string>mat_name
  DENSITY = <real>density_value
  #
  # thermal strain option
  THERMAL {LOG|ENGINEERING} STRAIN FUNCTION =
    <string>thermal_strain_function
  # or all three of the following
  THERMAL {LOG|ENGINEERING} STRAIN X FUNCTION =
    <string>thermal_strain_x_function
  THERMAL {LOG|ENGINEERING} STRAIN Y FUNCTION =
    <string>thermal_strain_y_function
  THERMAL {LOG|ENGINEERING} STRAIN Z FUNCTION =
    <string>thermal_strain_z_function
  #
  BEGIN PARAMETERS FOR MODEL FIBER_MEMBRANE
    YOUNGS MODULUS = <real>youngs_modulus
    POISSONS RATIO = <real>poissons_ratio
    SHEAR MODULUS = <real>shear_modulus
    BULK MODULUS = <real>bulk_modulus
    LAMBDA = <real>lambda
    CORD DENSITY = <real>cord_density
    CORD DIAMETER = <real>cord_diameter
    MATRIX DENSITY = <real>matrix_density
    TENSILE TEST FUNCTION = <string>test_function_name
    TEMPERATURE DEPENDENT TENSILE TEST FUNCTION =
       <string>temp_test_function_name
    PERCENT CONTINUUM = <real>percent_continuum
    EPL = <real>epl
    AXIS X = <real>axis_x
    AXIS Y = <real>axis_y
    AXIS Z = <real>axis_z
    MODEL = <string>RECTANGULAR|TIRE
    STIFFNESS SCALE = <real>stiffness_scale
    REFERENCE STRAIN = <real>reference_strain
    CORD YOUNGS MODULUS = <real>val
    CORD YOUNGS MODULUS TEMPERATURE FUNCTION = <string>val
    CORD YOUNGS MODULUS REGULARIZATION PARAMETER = <real>val
    MATRIX YOUNGS MODULUS = <real>val
    MATRIX YOUNGS MODULUS TEMPERATURE FUNCTION = <string>val
    CORD POISSONS RATIO = <real>val
    MATRIX POISSONS RATIO = <real>val
    POISSONS RATIO FUNCTION = <string>val
    CORD POISSONS RATIO FUNCTION = <string>val
    HALPIN TSAI HOMOGENIZATION = NONE | LINEAR | NONLINEAR
    MEMBRANE HOURGLASS STIFFNESS REFERENCE = COMPRESSION | TENSION
    INITIALIZATION TEMPERATURE = <real>initTemp
    INITIAL 3D MATERIAL STIFFNESS = SHEAR_MODULUS|ACTUAL
  END [PARAMETERS FOR MODEL FIBER_MEMBRANE]
END [MATERIAL <string>mat_name]
```

The fiber membrane model is used for modeling membranes that are reinforced with unidirectional fibers. Through the use of a non-zero `PERCENT CONTINUUM`, a background isotropic material response can be added such that the response in the fiber direction is unchanged. The fiber membrane model can be used in both Presto and Adagio. When the fiber membrane model is used in Adagio, the model can be used with or without the control-stiffness option in Adagio's multilevel solver. The control-stiffness option is implemented via the `CONTROL STIFFNESS` command block and is discussed in Chapter 2. If the control-stiffness option is activated in Adagio, the response in the fiber direction is softened by lowering the fiber response. In all cases, the final material behavior that is used for equilibrium corresponds to the real material response. When the fiber membrane model is used in Presto, the fiber scaling, which is controlled by the `STIFFNESS SCALE` command line, is ignored.

The command block for a fiber membrane material starts with the line:

```
BEGIN PARAMETERS FOR MODEL FIBER_MEMBRANE
```

and terminates with the line:

```
END [PARAMETERS FOR MODEL FIBER_MEMBRANE]
```

In the above command blocks, the following definitions are applicable. Usage requirements are identified both in this list of definitions and in the discussion that follows the list.

- The density of the material is defined with the `DENSITY` command line. This command line should be included, but its value will be recomputed (and hence replaced) if the `CORD DENSITY`, `CORD DIAMETER`, and `MATRIX DENSITY` command lines are specified.
- The `THERMAL LOG|ENGINEERING STRAIN X|Y|Z FUNCTION` command is used to define thermal strains. See Section 2 and Section 2 for further information on defining and activating thermal strains.
- Any two of the following elastic constants are required—these are used to compute values for the elastic preconditioner only.
  - Young's modulus E is defined with the `YOUNGS MODULUS` command line.
  - Poisson's ratio ν is defined with the `POISSONS RATIO` command line.
  - The bulk modulus K is defined with the `BULK MODULUS` command line.
  - The shear modulus μ is defined with the `SHEAR MODULUS` command line.
  - Lamé's first parameter λ is defined with the `LAMBDA` command line.
- The density and diameter of the fibers are defined by the `CORD DENSITY` and `CORD DIAMTER` command lines, respectively. These command lines are optional. See the usage discussion below.
- The density of the matrix is defined by the `MATRIX DENSITY` command line. This command line is optional. See the usage discussion below.

- The `TENSILE TEST FUNCTION` command line references the name of a function defined in a `FUNCTION` command line in the SIERRA scope that describes the fiber force versus strain data. This command line must be included.
- The `TEMPERATURE DEPENDENT TENSILE TEST FUNCTION` supplies a function name for the temperature dependent tensile test data. The x-value of this function specifies a temperature. The corresponding y-value specifies a unique integer function ID of a function that holds the strain versus force data at the x-value temperature. If the unique integer function ID is, for example, 2, then the strain versus force function is expected to have the name function_2.
- The fractional fiber stiffness to use in defining the background isotropic response is given by the `PERCENT CONTINUUM` command line. This command line must be included.
- The number of fibers per unit length is defined by the `EPL` command line. This command line must be included.
- The components of the vector defining the initial fiber direction are given by the `AXIS X`, `AXIS Y`, and `AXIS Z` command lines. These command lines must be included. See the usage discussion below.
- The coordinate system for specifying the fiber orientation is given by the `MODEL` command line. This command line must be included. See the usage discussion below.
- The fiber scaling is specified by the `STIFFNESS SCALE` command line. If the control-stiffness option is used in Adagio, this command line must be included. When the fiber membrane model is used in Presto, this command line is ignored.
- The reference strain is defined with the `REFERENCE STRAIN` command line. This command line is optional for Adagio and is not used in Presto. If the control-stiffness option is used in Adagio, this command line may be included. See the usage discussion below.

Certain command lines in the `PARAMETERS FOR MODEL FIBER_MEMBRANE` command block also have inter-dependencies or other factors that may impact their usage in Presto and Adagio, as discussed below.

The `CORD DENSITY`, `CORD DIAMETER`, and `MATRIX DENSITY` command lines are optional. When included, these three command lines are used for computation of the correct density corresponding to the fibers, the number of fibers per unit length, and the chosen matrix. When these three command lines are not included, the density is taken as that specified by the `DENSITY` command line.

The `AXIS X`, `AXIS Y`, and `AXIS Z` command lines must be specified if the value for the `MODEL` command line is `RECTANGULAR`.

Specifying `REFERENCE STRAIN` implies the use of strains for measuring part of the control-stiffness material constraint violation in Adagio. If this command line is not present, the material constraint violation is determined by comparing the change in the scaled fiber force over the current model problem.

4.2 Fiber Shell Model

```
BEGIN MATERIAL <string>mat_name
  DENSITY = <real>density_value
  #
  # thermal strain option
  THERMAL {LOG|ENGINEERING} STRAIN FUNCTION =
    <string>thermal_strain_function
  # or all three of the following
  THERMAL {LOG|ENGINEERING} STRAIN X FUNCTION =
    <string>thermal_strain_x_function
  THERMAL {LOG|ENGINEERING} STRAIN Y FUNCTION =
    <string>thermal_strain_y_function
  THERMAL {LOG|ENGINEERING} STRAIN Z FUNCTION =
    <string>thermal_strain_z_function
  #
  BEGIN PARAMETERS FOR MODEL FIBER_SHELL
    YOUNGS MODULUS = <real>youngs_modulus
    POISSONS RATIO = <real>poissons_ratio
    SHEAR MODULUS = <real>shear_modulus
    BULK MODULUS = <real>bulk_modulus
    LAMBDA = <real>lambda
    CORD DENSITY = <real>cord_density
    CORD DIAMETER = <real>cord_diameter
    MATRIX DENSITY = <real>matrix_density
    TENSILE TEST FUNCTION = <string>test_function_name
    PERCENT CONTINUUM = <real>percent_continuum
    EPL = <real>epl
    AXIS X = <real>axis_x
    AXIS Y = <real>axis_y
    AXIS Z = <real>axis_z
    MODEL = <string>RECTANGULAR
    ALPHA1 = <real>alpha1
    ALPHA2 = <real>alpha2
    ALPHA3 = <real>alpha3
    STIFFNESS SCALE = <real>stiffness_scale
    REFERENCE STRAIN = <real>reference_strain
    ORTHOTROPIC TRANSVERSE SHEAR H11 SCALE FACTOR =
      <real>h11_scale
    ORTHOTROPIC TRANSVERSE SHEAR H12 SCALE FACTOR =
      <real>h12_scale
    ORTHOTROPIC TRANSVERSE SHEAR H22 SCALE FACTOR =
      <real>h22_scale
    HALPIN TSAI HOMOGENIZATION = NONE | LINEAR | NONLINEAR |
      LINEAR KIRCHHOFF | NONLINEAR KIRCHHOFF
    BENDING CORD MODULUS <real>val
    BENDING CORD YOUNGS MODULUS TEMPERATURE FUNCTION =
      <string>func
      BENDING TEST FUNCTION = <string>func
      TEMPERATURE DEPENDENT BENDING TEST FUNCTION = <string>func
      BENDING CORD YOUNGS MODULUS FROM 3D MATERIAL = OFF|ON
    END [PARAMETERS FOR MODEL FIBER_SHELL]
  END [MATERIAL <string>mat_name]
```

The fiber shell model is used for modeling shells that are reinforced with unidirectional fibers. The input parameters for this model are generally identical to those for the fiber membrane described in Section 4.1; its membrane response should be nearly identical to that of the fiber membrane. The salient difference between the fiber membrane and shell models is that the latter includes bending and torsional stiffness contributions from the fiber thickness, whereas the former neglects these effects, yielding a two-dimensional physical response. The additional shell stiffnesses for the fiber shell model can be modified via the three input parameters `ALPHA1`, `ALPHA2`, and `ALPHA3`, as described below.

The fiber shell model currently only works with the analytically-integrated four-node shell element. For this element, it is essential to allocate space for storage of internal material parameters, and this is accomplished by requiring the specification of a single integration point for the shell. The thickness of the shell element is taken to be identical to the cord diameter, and this consistency condition should be explicitly provided in the input data. All other relevant shell element data fields, e.g., drilling stiffness factors, can be used as they would be for a conventional material model, e.g., isotropic linear elasticity.

The command block for a fiber shell material starts with the line

```
  BEGIN PARAMETERS FOR MODEL FIBER_SHELL
```

and terminates with the line

```
  END [PARAMETERS FOR MODEL FIBER_SHELL]
```

In the above command blocks, the various material definitions are identical to those specified for a fiber membrane material. The additional effects of plate-bending response are mediated by the three parameters `ALPHA1`, `ALPHA2`, and `ALPHA3`, and these three variant parameters are described below.

- The parameters `ALPHA1`, `ALPHA2`, and `ALPHA3` provide a means to modify the plate-bending stiffness computed by a fiber shell element, so that a factored bending response can be obtained. For example, to attenuate the effects of plate bending and torsion, these three parameters could be set to small values such as 0.001, to capture only the membrane response of the material. In practice, setting these parameters to such small values can degrade the conditioning of the finite element equation set, and setting them to zero risks generating a singular stiffness matrix, but the use of these three parameters can be deduced from this simple example. Setting all three of these parameters to 1.0 provides a consistent computation of plate bending and plate membrane effects.

  - The `ALPHA1` command line provides a factor multiplying the bending response of the shell in the fiber direction. Setting this parameter equal to unity captures all the bending response of the fiber shell in the fiber direction.
  - The `ALPHA2` command line provides a factor multiplying the bending response of the shell in the perpendicular-to-fiber direction. Setting this parameter equal to unity captures all the bending response of the fiber shell in the direction parallel to the plane of the shell, but perpendicular to the fiber direction.
  - The `ALPHA3` command line provides a factor multiplying the plate torsional response of the shell. Setting this parameter equal to unity captures all the relevant out-of-plane torsional response.
- The `ORTHOTROPIC TRANSVERSE SHEAR` H11, H12, and H22 factors scale the respective coefficients of the H-matrix for orthotropic bending.

Certain command lines in the `PARAMETERS FOR MODEL FIBER_SHELL` command block have interdependencies; additional details are available in the description of the fiber membrane model in Section 4.1.

4.3 Fiber Truss Model

This model exists buy has no documentation at this time

```
BEGIN MATERIAL FIBER TRUSS
  ...
END
```

Chapter 5

Elements

This chapter covers Goodyear specific element definitions.

5.1 Shell Section

```
BEGIN SHELL SECTION <string>shell_section_name
  THICKNESS = <real>shell_thickness
  THICKNESS MESH VARIABLE =
    <string>THICKNESS|<string>var_name
  THICKNESS TIME STEP = <real>time_value
  THICKNESS SCALE FACTOR = <real>thick_scale_factor(1.0)
  INTEGRATION RULE = TRAPEZOID|GAUSS|LOBATTO|SIMPSONS|
    USER|ANALYTIC|CODE_SELECTED(CODE_SELECTED)
  NUMBER OF INTEGRATION POINTS = <integer>num_int_points(5)
  FORMULATION = KH_SHELL|BT_SHELL|NQUAD(BT_SHELL)
  BEGIN USER INTEGRATION RULE
    <real>location_1 <real>weight_1
    <real>location_2 <real>weight_2
    .
    .
    <real>location_n <real>weight_n
  END [USER INTEGRATION RULE]
  LOFTING FACTOR = <real>lofting_factor(0.5)
  OFFSET MESH VARIABLE = <string>var_name
  ORIENTATION = <string>orientation_name
  DRILLING STIFFNESS FACTOR = <real>stiffness_factor(1.0e-5)
  RIGID BODY = <string>rigid_body_name
  RIGID BODIES FROM ATTRIBUTES = <integer>first_id
    TO <integer>last_id
END [SHELL SECTION <string>shell_section_name]
```

The `SHELL SECTION` command block is used to specify the properties for a shell element. If this command block is referenced in an element block of three-dimensional, four-node elements, the elements in the block will be treated as shell elements. The parameter, `shell_section_name`, is user-defined and is referenced by a `SECTION` command line in a `PARAMETERS FOR BLOCK` command block.

Either a `THICKNESS` command line or a `THICKNESS MESH VARIABLE` command line must appear in the `SHELL SECTION` command block.

If a shell element block references a `SHELL SECTION` command block with the command line:

```
THICKNESS = <real>shell_thickness
```

then all the shell elements in the block will have their thickness initialized to the value `shell_thickness`.

Sierra/SM can also initialize the thickness using an attribute defined on elements in the mesh file. Meshing programs such as PATRAN and Cubit typically set the element thickness as an attribute on the elements. If the elements have one and only one attribute defined on the mesh, the `THICKNESS MESH VARIABLE` command line should be specified as:

```
THICKNESS MESH VARIABLE = THICKNESS
```

which causes the thickness of the element to be initialized to the value of the attribute for that element. If there are zero attributes or more than one attribute, the thickness variable will not be automatically defined, and the command will fail.

The thickness may also be initialized by any other field present on the input mesh. To specify a field other than the single-element attribute, use this form of the `THICKNESS MESH VARIABLE` command line:

```
  THICKNESS MESH VARIABLE = <string>var_name
```

Here, the string `var_name` is the name of the initializing field.

The input mesh may have values defined at more than one point in time. To choose the point in time in the mesh file that the variable should be read, use the command line:

```
  THICKNESS TIME STEP = <real>time_value
```

The default time point in the mesh file at which the variable is read is 0.0.

Once the thickness of a shell element is initialized by using either the `THICKNESS` command line or the `THICKNESS MESH VARIABLE` command line, this initial thickness value can then be scaled using the scale-factor command line:

```
  THICKNESS SCALE FACTOR = <real>thick_scale_factor
```

If the initial thickness of the shell is 0.15 inch, and the value for `thick_scale_factor` is 0.5, then the scaled thickness of the membrane will be 0.075.

The thickness mesh variable specification may be coupled with the `THICKNESS SCALE FACTOR` command line. In this case, the thickness mesh variable is scaled by the specified factor.

The shell formulation can be selected via the `FORMULATION` command line. Several options are available:

- The `BT_SHELL` option is used to select the Belytschko-Tsay (BT) shell formulation. This is the default. The BT shell formulation has constant transverse shear, and is the fastest shell element available.
- The `KH_SHELL` option specifies the the Key-Hoff shell formulation, which includes a nonlinear transverse shear term. The Key-Hoff shell can better solve problems with high gradients of transverse shear (such as twisted beams), but distorted elements can be excessively stiff.
- The `NQUAD` option selects a formulation for a completely elastic plate.
- The `SEVEN_PARAMETER` option selects an experimental shell formulation.
- The `THICKSHELL` option selects an experimental shell element that treats hexahedral elements with a shell-based formulation.

For shell elements, the user can select from a number of integration rules, including a user-defined integration option. The integration rule is selected with the command line:

```
  INTEGRATION RULE =
<string>TRAPEZOID|GAUSS|LOBATTO|SIMPSONS|ANALYTIC|CODE_SELECTED
      USER(CODE_SELECTED)
```

Consult the element documentation [1] for a description of different integration schemes for shell elements.

For most material models the code will by default select the five point TRAPEZOID through the thickness integration rule. The code selected integration scheme for the elastic material model and the fiber shell material model is ANALYTIC, which is an analytic through-thickness integration scheme. This scheme is significantly faster and slightly more accurate than any of the point-wise rules but only valid for fully elastic shell formulations.

Shells using using the analytic integration rule do not compute integration point material or stress quantities such as stress, strain, or the various quantities derived from stress and strain. If integration point quantities are desired for elastic material shells, the shells can be forced to use a point-wise integration scheme by manually setting the integration rule to TRAPEZOID or another point-wise rule.

Currently the analytic integration scheme is only available for the elastic material model when using the BT_SHELL shell formulation for quad shell elements, and for the fiber shell material model using the BT_SHELL formulation with one integration point allocated for storage of the inelastic fiber material parameters.

The number of integration points for the piecewise integration point rules can be set to any number greater than one by using the following command line:

```
NUMBER OF INTEGRATION POINTS = <integer>num_int_points(5)
```

The SIMPSONS, GAUSS, and LOBATTO integration schemes in the INTEGRATION RULE command line all default to five integration points. The number of integration points for these three schemes can be reset by using the NUMBER OF INTEGRATION POINTS command line. There are limitations on the number of integration points for some of these integration rules. The SIMPSONS rule can be set to any number greater than one, the GAUSS scheme can be set to one through seven integration points, and the LOBATTO integration scheme can be set to two through seven integration points.

In addition to these standard integration schemes, you may also define an integration scheme by using the USER INTEGRATION RULE command block.

```
BEGIN USER INTEGRATION RULE
  <real>location_1 <real>weight_1
  <real>location_2 <real>weight_2
  .
  .
  <real>location_n <real>weight_n
END [USER INTEGRATION RULE]
```

You may NOT specify both a standard integration scheme and a user scheme. If the USER option is specified in the INTEGRATION RULE command line, a set of integration locations with associated weight factors must be specified. This is done with tabular input command lines inside the USER INTEGRATION RULE command block. The number of command lines inside this command block should match the number of integration points specified in the NUMBER OF INTEGRATION POINTS command line. For example, suppose we wish to use a user-defined scheme with three integration points. The NUMBER OF INTEGRATION POINTS command line should specify three

(3) integration points and the number of command lines inside the `USER INTEGRATION RULE` command block should be three (to give three locations and three weight factors).

For the user-defined rule, the integration point locations should fall between –1 and +1, and the weights should sum to 1.0.

The command line

```
  LOFTING FACTOR = <real>lofting_factor(0.5)
```

allows the user to shift the location of the mid-surface of a shell element relative to the geometric location of the shell element. By default, the geometric location of a shell element in a mesh represents the mid-surface of the shell. If a shell has a thickness of 0.2 inch, the top surface of the shell is 0.1 inch above the geometric surface defined by the shell element, and the bottom surface of the shell is 0.1 inch below the geometric surface defined by the shell element. (The top surface of the shell is the surface with a positive element normal; the bottom surface of the shell is the surface with a negative element normal.)

Figure 5.1 shows an edge-on view of shell elements with a thickness of *t* and the location of the geometric plane in relation to the shell surfaces for three different values of the lofting factor—0.0, 0.5, and 1.0. For a lofting factor of 0.0, the geometric surface defined by the shell corresponds to the top surface of the shell element. A lofting factor of 1.0 puts the geometric surface at the bottom surface of the shell element. The geometric surface is midway between the top and bottom surfaces for a lofting factor of 0.5, which is the default. Lofting factors greater than 1.0 or less than 0.0 is typically reserved for layering of shells otherwise this is a bad idea.

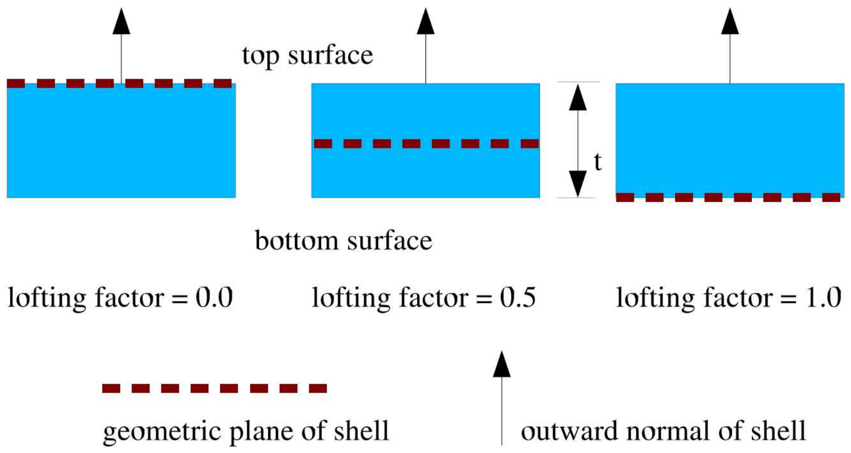

Figure 5.1: Location of geometric plane of shell for various lofting factors.

Consider the example of a lofting factor set to 1.0 for a shell with thickness of 0.2 inch. In this case, the top surface of the shell will be located at a distance of 0.2 inch from the geometric surface (measuring in the direction of the positive shell normal), and the bottom surface will be located at the geometric surface.

Both the shell mechanics and contact use shell lofting. See Section 2 for a discussion of lofting surfaces for shells and contact. It is recommended that shell lofting values other than 0.5 not be used if the shell is thick. If the shell is thicker than its in-plane width, the shell lofting algorithms may become unstable.

Lofting as described above may also be implemented through

```
OFFSET MESH VARIABLE = <string>var_name.
```

This command allows an offset value to be read from an attribute (or variable) on the mesh file. This attribute (or variable) must be named but may have any name, and this name is specified in place of `var_name` in the command. An offset is a dimensional (i.e., not scaled by the shell thickness) value that gives the shell mid-plane shift in the positive shell normal direction. Internal to Sierra/SM the offset value is converted to an equivalent lofting factor by dividing by the initial thickness and adding 0.5. Thus, an offset of zero gives a lofting factor of 0.5, an offset of one half of the thickness gives a lofting factor of one, and an offset of negative one half of the thickness gives a lofting factor of zero. There is no check that given offset values produce lofting values between zero and one, which allows any offset to be specified but requires that care be exercised to avoid unstable lofted shells.

Warning: An offset and a lofting factor may not be specified in the same shell section. Both determine the shell lofting and may conflict.

The `ORIENTATION` command line lets you select a coordinate system for output of in-plane stresses and strains. The `ORIENTATION` option makes use of an embedded coordinate system *rst* associated with each shell element. The *rst* coordinate system for a shell element is shown in Figure 5.2. The *r*-axis extends from the center of the shell to the midpoint of the side of the shell defined by nodes 1 and 2. The *t*-axis is located at the center of the shell and is normal to the surface of the shell at the center point. The *s*-axis is the cross-product of the *t*-axis and the *r*-axis. The *rst*-axes form a local coordinate system at the center of the shell; this local coordinate system moves with the shell element as the element deforms.

The `ORIENTATION` command line in the `SHELL SECTION` command block references an `ORIENTATION` command block that appears in the SIERRA scope. As described in Chapter 2 of this document, the `ORIENTATION` command block can be used to define a local co-rotational coordinate system $X''Y''Z''$ at the center of a shell element. In the original shell configuration (time 0), one of the axes—X'', Y'', or Z''—is projected onto the plane of the shell element. The angle between this projected axis of the $X''Y''Z''$ coordinate system and the *r*-axis is used to establish the transformation for in-plane stresses and strains. We will illustrate this with an example.

Suppose that in our `ORIENTATION` command block we have specified a rotation of 30 degrees about the 1-axis (X'-axis). The command line for this rotation in the `ORIENTATION` command block would be:

```
ROTATION ABOUT 1 = 30
```

For this case, we project the Y''-axis onto the plane of the shell (Figure 5.3). The angle between this projection and the *r*-axis establishes a transformation for the in-plane stresses of the shell (the stresses in the center of the shell lying in the plane of the shell). What will be output as the in-plane stress σ^{ip}_{xx} will be in the Y''-direction; what will be output as the in-plane stress σ^{ip}_{yy} will be in the Z''-direction. The in-plane stress σ^{ip}_{xy} is a shear stress in the $Y''Z''$-plane. The $X''Y''Z''$ coordinate

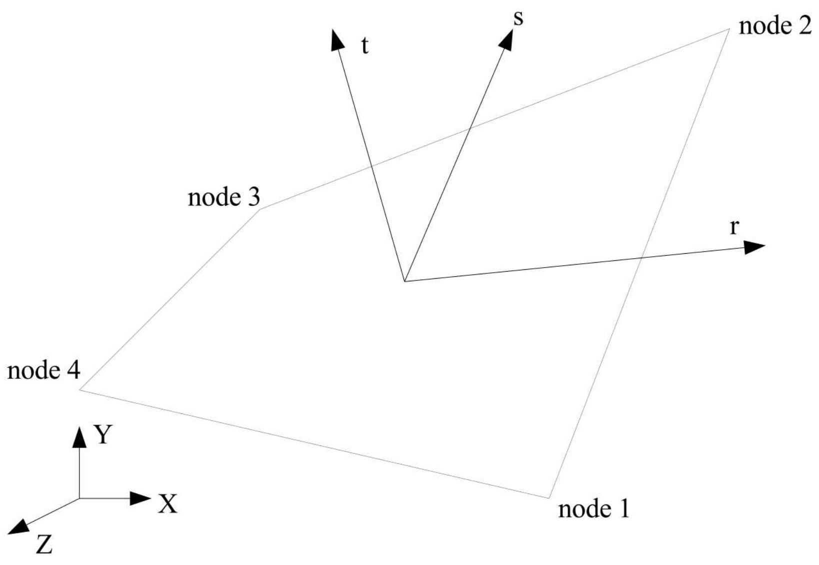

Figure 5.2: Local *rst* coordinate system for a shell element.

system maintains the same relative position in regard to the *rst* coordinate system. This means that the $X''Y''Z''$ coordinate system is a local coordinate system that moves with the shell element as the element deforms.

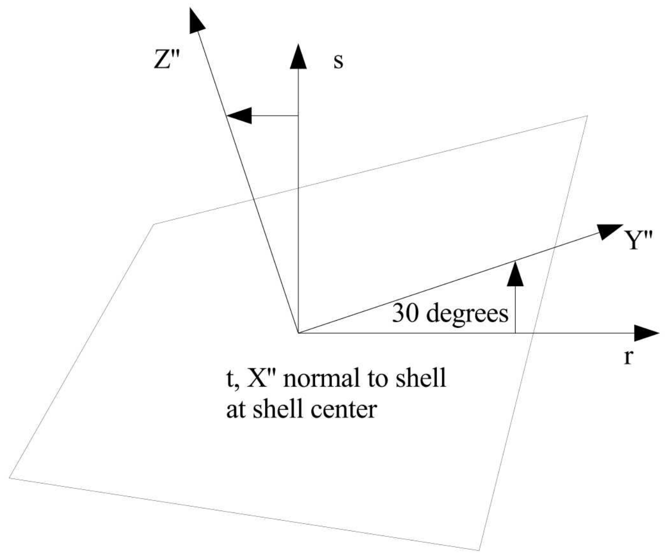

Figure 5.3: Rotation of 30 degrees about the 1-axis (X'-axis).

The following permutations for output of the in-plane stresses occur depending on the axis (1, 2, or 3) specified in the `ROTATION ABOUT` command line:

- Rotation about the 1-axis (X'-axis): The in-plane stress σ^{ip}_{xx} will be in the Y''-direction; the in-plane stress σ^{ip}_{yy} will be in the Z''-direction. The in-plane stress σ^{ip}_{xy} is a shear stress in the $Y''Z''$-plane.

- Rotation about the 2-axis (Y'-axis): The in-plane stress σ^{ip}_{xx} will be in the Z''-direction; the in-plane stress σ^{ip}_{yy} will be in the X''-direction. The in-plane stress σ^{ip}_{xy} is a shear stress in the $Z''X''$-plane.
- Rotation about the 3-axis (Z'-axis): The in-plane stress σ^{ip}_{xx} will be in the X''-direction; the in-plane stress σ^{ip}_{yy} will be in the Y''-direction. The in-plane stress σ^{ip}_{xy} is a shear stress in the $X''Y''$-plane.

If no orientation is specified, the in-plane stresses and strains are output in the consistent axes from a default orientation, which is a rectangular system with the X' and Y' axes taken as the global X and Y axes, respectively, and `ROTATION ABOUT 1 = 0.0`.

The command line

```
DRILLING STIFFNESS FACTOR = <real>stiffness_factor
```

adds stiffness in the drilling degrees of freedom to quadrilateral shells. Drilling degrees of freedom are rotational degrees of freedom in the direction orthogonal to the plane of the shell at each node. The formulation used for the quadrilateral shells has no rotational stiffness in this direction. This can lead to spurious zero-energy modes of deformation similar in nature to hourglass deformation. This makes obtaining a solution difficult in quasistatic problems and can result in singularities when using the full tangent preconditioner.

The `stiffness_factor` should be chosen as a quantity small enough to add enough stiffness to allow the solve to be successful without unduly affecting the solution. The default value for `stiffness_factor` is 1.0e-5, which places drilling stiffness about 10000X lower than the bending stiffness. If singularities are encountered in the solution or hourglass-like deformation is observed in the drilling degrees of freedom, setting a larger value may be appropriate.

Elements in a section can be made rigid by including the `RIGID BODY` command line. The `RIGID BODY` command line specifies an indenter that maps to a rigid body command block. See Section 2 for a full discussion of how to create rigid bodies and Section 2 for information on the use of the `RIGID BODIES FROM ATTRIBUTES` command.

References

[1] T.A. Laursen, S.W. Attaway, and R.I. Zadoks. SEACAS theory manuals: Part III. finite element analysis in nonlinear solid mechanics. Technical Report SAND98-1760/3, Sandia National Laboratories, Albuquerque, NM, 1999. pdf.

Chapter 6

Contact

This chapter describes Goodyear specific sections of contact.

Implicit Only

6.1 Legacy Contact

This section describes the use of the contact library written for JAS3D. This library is called the Legacy Contact Library or LCLib.

Adagio has two contact search and enforcement options. The first option, the default, is to use ACME for the contact search and Adagio's own enforcement library. The second option is to use JAS3D's contact library for both the search and the enforcement. While the capabilities of the two options are nominally the same, one may perform better than the other for certain problems.

To invoke the use of LCLib, include the line command

```
JAS MODE
```

in the region.

When using LCLib, only one `CONTACT DEFINITION` block is allowed in the region. The `ENFORCEMENT` line command must invoke the `FRICTIONAL` type. If frictionless contact is desired for an interaction, set the `FRICTION COEFFICIENT` to zero. If tied contact is desired for an interaction, set the `FRICTION COEFFICIENT` to -1.

Index

Distribution

1	0899	Technical Library, 9536 (1 electronic)